15

Haga lo que hicimos: autorización de solicitudes

¡Ahora está contigo! Realice el mismo procedimiento que hice en clase, implementando los códigos necesarios para realizar el control de acceso en la API.

Opinión del instructor

Deberá crear una clase de `Filter`, responsable de interceptar solicitudes y realizar el proceso de autenticación y autorización:

```
@Component
public class SecurityFilter extends OncePerRequestFilter {

    @Autowired
    private TokenService tokenService;

    @Autowired
    private UsuarioRepository repository;

    @Override
    protected void doFilterInternal(HttpServletRequest request
        var tokenJWT = recuperarToken(request);

        if (tokenJWT != null) {
            var subject = tokenService.getSubject(tokenJWT);
            var usuario = repository.findByLogin(subject);
```

```
                var authentication = new UsernamePasswordAuthentic
                SecurityContextHolder.getContext().setAuthenticati
            }

            filterChain.doFilter(request, response);
        }

        private String recuperarToken(HttpServletRequest request)
            var authorizationHeader = request.getHeader("Authoriza
            if (authorizationHeader != null) {
                return authorizationHeader.replace("Bearer ", "");
            }

            return null;
        }

}
```

COPIA EL CÓDIGO

También deberá actualizar el código de la clase `SecurityConfigurations`:

```
@Configuration
@EnableWebSecurity
public class SecurityConfigurations {

    @Autowired
    private SecurityFilter securityFilter;

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurit
        return http.csrf().disable()
                .sessionManagement().sessionCreationPolicy(Ses
```

```
                .and().authorizeRequests()
                .antMatchers(HttpMethod.POST, "/login").permit
                .anyRequest().authenticated()
                .and().addFilterBefore(securityFilter, Usernam
                .build();
    }

    @Bean
    public AuthenticationManager authenticationManager(Authent
        return configuration.getAuthenticationManager();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

}
```

COPIA EL CÓDIGO

Y finalmente, deberá actualizar el código de la clase `TokenService` :

```
@Service
public class TokenService {

    @Value("${api.security.token.secret}")
    private String secret;

    public String generarToken(Usuario usuario) {
        try {
            var algoritmo = Algorithm.HMAC256(secret);
            return JWT.create()
```

```
                    .withIssuer("API Voll.med")
                    .withSubject(usuario.getLogin())
                    .withExpiresAt(fechaExpiracion())
                    .sign(algoritmo);
        } catch (JWTCreationException exception){
            throw new RuntimeException("error al generar token
        }
    }

    public String getSubject(String tokenJWT) {
        try {
            var algoritmo = Algorithm.HMAC256(secret);
            return JWT.require(algoritmo)
                    .withIssuer("API Voll.med")
                    .build()
                    .verify(tokenJWT)
                    .getSubject();
        } catch (JWTVerificationException exception) {
            throw new RuntimeException("Token JWT inválido o e
        }
    }

    private Instant fechaExpiracion() {
        return LocalDateTime.now().plusHours(2).toInstant(Zone
    }

}
```

COPIA EL CÓDIGO